

## Durham Research Online

---

### Deposited in DRO:

12 June 2019

### Version of attached file:

Published Version

### Peer-review status of attached file:

Peer-reviewed

### Citation for published item:

Klimošová, T. and Malík, J. and Masařík, T. and Novotná, J. and Paulusma, D. and Slívová, V. (2018) 'Colouring (Pr+Ps)-free graphs.', in 29th International Symposium on Algorithms and Computation (ISAAC 2018). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 5:1-5:13. Leibniz international proceedings in informatics. (123).

### Further information on publisher's website:

<https://doi.org/10.4230/LIPIcs.ISAAC.2018.5>

### Publisher's copyright statement:

© Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, Veronika Slívová; licensed under Creative Commons License CC-BY.

### Additional information:

---

### Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a [link](#) is made to the metadata record in DRO
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the [full DRO policy](#) for further details.

# Colouring $(P_r + P_s)$ -Free Graphs

**Tereza Klimošová**

Department of Applied Mathematics, Charles University, Prague, Czech Republic  
tereza@kam.mff.cuni.cz

**Josef Malík**

Czech Technical University in Prague, Czech Republic  
malikjo1@fit.cvut.cz

**Tomáš Masařík**

Department of Applied Mathematics, Charles University, Prague, Czech Republic  
masarik@kam.mff.cuni.cz

**Jana Novotná**

Department of Applied Mathematics, Charles University, Prague, Czech Republic  
janca@kam.mff.cuni.cz

**Daniël Paulusma**

Department of Computer Science, Durham University, Durham, UK  
daniel.paulusma@durham.ac.uk

**Veronika Slívová**

Computer Science Institute of Charles University, Prague, Czech Republic  
slivova@iuuk.mff.cuni.cz

---

## Abstract

The  $k$ -COLOURING problem is to decide if the vertices of a graph can be coloured with at most  $k$  colours for a fixed integer  $k$  such that no two adjacent vertices are coloured alike. If each vertex  $u$  must be assigned a colour from a prescribed list  $L(u) \subseteq \{1, \dots, k\}$ , then we obtain the LIST  $k$ -COLOURING problem. A graph  $G$  is  $H$ -free if  $G$  does not contain  $H$  as an induced subgraph. We continue an extensive study into the complexity of these two problems for  $H$ -free graphs. We prove that LIST 3-COLOURING is polynomial-time solvable for  $(P_2 + P_5)$ -free graphs and for  $(P_3 + P_4)$ -free graphs. Combining our results with known results yields complete complexity classifications of 3-COLOURING and LIST 3-COLOURING on  $H$ -free graphs for all graphs  $H$  up to seven vertices. We also prove that 5-COLOURING is NP-complete for  $(P_3 + P_5)$ -free graphs.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Graph theory

**Keywords and phrases** vertex colouring,  $H$ -free graph, linear forest

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2018.5

**Related Version** A preprint of this article is available on arXiv [22], <https://arxiv.org/abs/1804.11091v2>.

**Funding** T. Masařík, J. Novotná and V. Slívová were supported by the project GAUK 1277018 and the grant SVV-2017-260452. T. Klimošová was supported by the Center of Excellence – ITI, project P202/12/G061 of GA ČR, by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004), and by the project GAUK 1277018. T. Masařík was also partly supported by the Center of Excellence – ITI, project P202/12/G061 of GA ČR. V. Slívová was partly supported by the project 17-09142S of GA ČR and Charles University project PRIMUS/17/SCI/9. D. Paulusma was supported by the Leverhulme Trust (RPG-2016-258).

**Acknowledgements** We thank Karel Král for helpful comments.



© Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová;

licensed under Creative Commons License CC-BY

29th International Symposium on Algorithms and Computation (ISAAC 2018).

Editors: Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao; Article No. 5; pp. 5:1–5:13



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Graph colouring is a popular concept in Computer Science and Mathematics due to a wide range of practical and theoretical applications, as evidenced by numerous surveys and books on graph colouring and many of its variants (see, for example, [5, 14, 21, 24, 28, 30, 33]). Formally, a *colouring* of a graph  $G = (V, E)$  is a mapping  $c : V \rightarrow \{1, 2, \dots\}$  that assigns each vertex  $u \in V$  a *colour*  $c(u)$  in such a way that  $c(u) \neq c(v)$  whenever  $uv \in E$ . If  $1 \leq c(u) \leq k$ , then  $c$  is also called a  $k$ -colouring of  $G$  and  $G$  is said to be  $k$ -colourable. The COLOURING problem is to decide if a given graph  $G$  has a  $k$ -colouring for some given integer  $k$ .

It is well known that COLOURING is NP-complete even if  $k = 3$  [27]. To pinpoint the reason behind the computational hardness of COLOURING one may impose restrictions on the input. This led to an extensive study of COLOURING for special graph classes, particularly hereditary graph classes. A graph class is *hereditary* if it is closed under vertex deletion. As this is a natural property, hereditary graph classes capture a very large collection of well-studied graph classes. It is readily seen that a graph class  $\mathcal{G}$  is hereditary if and only if  $\mathcal{G}$  can be characterized by a unique set  $\mathcal{H}_{\mathcal{G}}$  of minimal forbidden induced subgraphs. If  $\mathcal{H}_{\mathcal{G}} = \{H\}$ , then a graph  $G \in \mathcal{G}$  is called  $H$ -free.

Král', Kratochvíl, Tuza, and Woeginger [23] started a systematic study into the complexity of COLOURING on  $\mathcal{H}$ -free graphs for sets  $\mathcal{H}$  of size at most 2. They showed polynomial-time solvability if  $H$  is an induced subgraph of  $P_4$  or  $P_1 + P_3$  and NP-completeness for all other graphs  $H$ . The classification for the case where  $\mathcal{H}$  has size 2 is far from finished; see the summary in [14] or an updated partial overview in [11] for further details. Instead of considering sets  $\mathcal{H}$  of size 2, we consider  $H$ -free graphs and follow another well-studied direction, in which the number of colours  $k$  is *fixed*, that is,  $k$  no longer belongs to the input.

$k$ -COLOURING: Given a graph  $G$  does there exist a  $k$ -colouring of  $G$ ?

A  $k$ -list assignment of  $G$  is a function  $L$  with domain  $V$  such that the *list of admissible colours*  $L(u)$  of each  $u \in V$  is a subset of  $\{1, 2, \dots, k\}$ . A colouring  $c$  *respects*  $L$  if  $c(u) \in L(u)$  for every  $u \in V$ . If  $k$  is fixed, then we obtain the following generalization of  $k$ -COLOURING:

LIST  $k$ -COLOURING: Given a graph  $G$  and a  $k$ -list assignment  $L$  does there exist a colouring of  $G$  that respects  $L$ ?

For every  $k \geq 3$ ,  $k$ -COLOURING on  $H$ -free graphs is NP-complete if  $H$  contains a cycle [13] or an induced claw [19, 26]. Hence, the case where  $H$  is a *linear forest* (a disjoint union of paths) remains. The situation is far from settled yet, although many partial results are known [2, 3, 4, 6, 7, 8, 9, 10, 15, 18, 20, 25, 29, 31, 34]. Particularly, the case where  $H$  is the  $t$ -vertex path  $P_t$  has been well studied. The cases  $k = 4$ ,  $t = 7$  and  $k = 5$ ,  $t = 6$  are NP-complete [20]. For  $k \geq 1$ ,  $t = 5$  [18] and  $k = 3$ ,  $t = 7$  [2], even LIST  $k$ -COLOURING on  $P_t$ -free graphs is polynomial-time solvable (see also [14]). For a fixed integer  $k$ , the  $k$ -PRECOLOURING EXTENSION problem is to decide a given  $k$ -colouring defined on an induced subgraph of a graph  $G$  can be extended to a  $k$ -colouring of  $G$ . Recently it was shown in [7, 8] that 4-PRECOLOURING EXTENSION, and therefore 4-COLOURING, is polynomial-time solvable for  $P_6$ -free graphs. In contrast, the more general problem LIST 4-COLOURING is NP-complete for  $P_6$ -free graphs [15]. See Table 1 for a summary of all these results.

From Table 1 we see that only the cases  $k = 3$ ,  $t \geq 8$  are still open, although some partial results are known for  $k$ -COLOURING for the case  $k = 3$ ,  $t = 8$  [9]. The situation when  $H$  is a disconnected linear forest  $\bigcup P_i$  is less clear. It is known that for every  $s \geq 1$ , LIST 3-COLOURING is polynomial-time solvable for  $sP_3$ -free graphs [4, 14]. For every graph  $H$ ,

■ **Table 1** Summary for  $P_t$ -free graphs.

$t$	$k$ -COLOURING				$k$ -PRECOLOURING EXTENSION				LIST $k$ -COLOURING			
	$k = 3$	$k = 4$	$k = 5$	$k \geq 6$	$k = 3$	$k = 4$	$k = 5$	$k \geq 6$	$k = 3$	$k = 4$	$k = 5$	$k \geq 6$
$t \leq 5$	P	P	P	P	P	P	P	P	P	P	P	P
$t = 6$	P	P	NP-c	NP-c	P	P	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t = 7$	P	NP-c	NP-c	NP-c	P	NP-c	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t \geq 8$	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c

LIST 3-COLOURING is polynomial-time solvable for  $(H + P_1)$ -free graphs if it is polynomially solvable for  $H$ -free graphs [4, 14]. If  $H = rP_1 + P_5$  ( $r \geq 0$ ) a stronger result is known.

► **Theorem 1** ([10]). *For all  $k \geq 1, r \geq 0$ , LIST  $k$ -COLOURING is polynomial-time solvable on  $(rP_1 + P_5)$ -free graphs.*

Theorem 1 cannot be extended to larger linear forests  $H$ , as LIST 4-COLOURING is NP-complete for  $P_6$ -free graphs [15] and LIST 5-COLOURING is NP-complete for  $(P_2 + P_4)$ -free graphs [10]. As mentioned, 5-COLOURING is known to be NP-complete for  $P_6$ -free graphs [20], but the existence of integers  $k \geq 3$  and  $2 \leq r \leq 5$  such that  $k$ -COLOURING is NP-complete for  $(P_r + P_5)$ -free graphs has not been shown in the literature.

Another way of making progress is to complete a classification by bounding the size of  $H$ . It follows from the above results and the ones in Table 1 that for a graph  $H$  with  $|V(H)| \leq 6$ , 3-COLOURING and LIST 3-COLOURING (and consequently, 3-PRECOLOURING EXTENSION) are polynomial-time solvable on  $H$ -free graphs if  $H$  is a linear forest, and NP-complete otherwise; see also [14]. In [14] it was also shown that, to obtain the same statement for graphs  $H$  with  $|V(H)| \leq 7$ , only the two cases where  $H \in \{P_2 + P_5, P_3 + P_4\}$  must be solved.

**Our Results.** In Section 2 we solve the two missing cases listed above.

► **Theorem 2.** *LIST 3-COLOURING is polynomial-time solvable for  $(P_2 + P_5)$ -free graphs and for  $(P_3 + P_4)$ -free graphs.*

We prove Theorem 2 as follows. If the graph  $G$  of an instance  $(G, L)$  of LIST 3-COLOURING is  $P_7$ -free, then we can use the aforementioned result of Bonomo et al. [2]. Hence we may assume that  $G$  contains an induced  $P_7$ . We consider every possibility of colouring the vertices of this  $P_7$  and try to reduce each resulting instance to a polynomial number of smaller instances of 2-SATISFIABILITY. As the latter problem can be solved in polynomial time, the total running time of the algorithm will be polynomial. The crucial proof ingredient is that we partition the set of vertices of  $G$  that do not belong to the  $P_7$  into subsets of vertices that are of the same distance to the  $P_7$ . This leads to several “layers” of  $G$ . We analyse how the vertices of each layer are connected to each other and to vertices of adjacent layers so as to use this information in the design of our algorithm.

Combining Theorem 2 with the aforementioned known results yields the following complexity classifications for graphs  $H$  up to seven vertices.

► **Corollary 3.** *Let  $H$  be a graph with  $|V(H)| \leq 7$ . If  $H$  is a linear forest, then LIST 3-COLOURING is polynomial-time solvable for  $H$ -free graphs; otherwise already 3-COLOURING is NP-complete for  $H$ -free graphs.*

In Section 3 we complement Theorem 2 by proving the following result.

► **Theorem 4.** *5-COLOURING is NP-complete for  $(P_3 + P_5)$ -free graphs.*

## Preliminaries

Let  $G = (V, E)$  be a graph. For a vertex  $v \in V$ , we denote its *neighbourhood* by  $N(v) = \{u \mid uv \in E\}$ , its *closed neighbourhood* by  $N[v] = N(v) \cup \{v\}$  and its degree by  $\deg(v) = |N(v)|$ . For a set  $S \subseteq V$ , we write  $N(S) = \bigcup_{v \in S} N(v) \setminus S$  and  $N[S] = N(S) \cup S$ , and we let  $G[S] = (S, \{uv \mid u, v \in S\})$  be the subgraph of  $G$  induced by  $S$ . The *contraction* of an edge  $e = uv$  removes  $u$  and  $v$  from  $G$  and introduces a new vertex which is made adjacent to every vertex in  $N(u) \cup N(v)$ . The *identification* of a set  $S \subseteq V$  by a vertex  $w$  removes all vertices of  $S$  from  $G$ , introduces  $w$  as a new vertex and makes  $w$  adjacent to every vertex in  $N(S)$ . The *length* of a path is its number of edges. The *distance*  $\text{dist}_G(u, v)$  between two vertices  $u$  and  $v$  is the length of a shortest path between them in  $G$ . The *distance*  $\text{dist}_G(u, S)$  between a vertex  $u \in V$  and a set  $S \subseteq V \setminus \{v\}$  is defined as  $\min\{\text{dist}(u, v) \mid v \in S\}$ .

For two graphs  $G$  and  $H$ , we use  $G + H$  to denote the disjoint union of  $G$  and  $H$ , and we write  $rG$  to denote the disjoint union of  $r$  copies of  $G$ . Let  $(G, L)$  be an instance of LIST 3-COLOURING. For  $S \subseteq V(G)$ , we write  $L(S) = \bigcup_{u \in S} L(u)$ . We let  $P_n$  and  $K_n$  denote the path and complete graph on  $n$  vertices, respectively. The *diamond* is the graph obtained from  $K_4$  after removing an edge. We say that an instance  $(G', L')$  is *smaller* than some other instance  $(G, L)$  of LIST 3-COLOURING if either  $G'$  is an induced subgraph of  $G$  with  $|V(G')| < |V(G)|$ ; or  $G' = G$  and  $L'(u) \subseteq L(u)$  for each  $u \in V(G)$ , such that there exists at least one vertex  $u^*$  with  $L'(u^*) \subset L(u^*)$ .

## 2 The Two Polynomial-Time Results

In this section we show that LIST 3-COLOURING problem is polynomial-time solvable for  $(P_2 + P_5)$ -free graphs and for  $(P_3 + P_4)$ -free graphs. As arguments for these two graph classes are overlapping, we prove both cases simultaneously. Our proof uses the following two results.

► **Theorem 5** ([2]). LIST 3-COLOURING is polynomial-time solvable for  $P_7$ -free graphs.

► **Theorem 6** ([12]). The 2-LIST COLOURING problem is linear-time solvable.

**Outline of the proof of Theorem 2.** Our goal is to reduce, in polynomial time, an instance  $(G, L)$  of LIST 3-COLOURING, where  $G$  is  $(P_2 + P_5)$ -free or  $(P_3 + P_4)$ -free, to a polynomial number of smaller instances of 2-LIST-COLOURING in such a way that  $(G, L)$  is a yes-instance if and only if at least one of the new instances is a yes-instance. As for each of the smaller instances, we can apply Theorem 6, the total running time of our algorithm will be polynomial.

If  $G$  is  $P_7$ -free, then we do not have to do the above and may apply Theorem 5 instead. Hence, we assume that  $G$  contains an induced  $P_7$ . We put the vertices of the  $P_7$  in a set  $N_0$  and define sets  $N_i$  ( $i \geq 1$ ) of vertices of the same distance  $i$  from  $N_0$ ; we say that the sets  $N_i$  are the layers of  $G$ . We then analyse the structure of these layers using the fact that  $G$  is  $(P_2 + P_5)$ -free or  $(P_3 + P_4)$ -free. The first phase of our algorithm is about preprocessing  $(G, L)$  after colouring the seven vertices of  $N_0$  and applying a number of propagation rules. We consider every possible colouring of the vertices of  $N_0$ . In each branch we may have to deal with vertices  $u$  that still have a list  $L(u)$  of size 3. We call such vertices active and prove that they all belong to  $N_2$ . We then enter the second phase of our algorithm. In this phase we show, via some further branching, that  $N_1$ -neighbours of active vertices either all have a list from  $\{\{h, i\}, \{h, j\}\}$ , where  $\{h, i, j\} = \{1, 2, 3\}$ , or they all have the same list  $\{h, i\}$ . In the third phase we reduce, again via some branching, to the situation where only the latter option applies:  $N_1$ -neighbours of active vertices all have the same list. Then in the

fourth and final phase of our algorithm we know so much structure of the instance that we can reduce to a polynomial number of smaller instances of 2-LIST-COLOURING via a new propagation rule identifying common neighbourhoods of two vertices by a single vertex.

► **Theorem 2 (restated).** LIST 3-COLOURING is polynomial-time solvable for  $(P_2 + P_5)$ -free graphs and for  $(P_3 + P_4)$ -free graphs.

**Proof Sketch.** Due to space limitation we omit the proof for the (more involved) case where  $H = P_3 + P_4$ . Hence, let  $(G, L)$  be an instance of LIST 3-COLOURING, where  $G = (V, E)$  is a  $(P_2 + P_5)$ -free graph. Whenever possible, we base our arguments on  $(P_3 + P_5)$ -freeness. Since the problem can be solved component-wise, we may assume that  $G$  is connected. If  $G$  contains a  $K_4$ , then  $G$  is not 3-colourable, and thus  $(G, L)$  is a no-instance. As we can decide if  $G$  contains a  $K_4$  in  $O(n^4)$  time by brute force, we assume that from now on  $G$  is  $K_4$ -free. By brute force we either deduce in  $O(n^7)$  time that  $G$  is  $P_7$ -free or we find an induced  $P_7$  on vertices  $v_1, \dots, v_7$  in that order. In the first case we use Theorem 5. It remains to deal with the second case.

► **Definition 7 (Layers).** Let  $N_0 = \{v_1, \dots, v_7\}$ . For  $i \geq 1$ , we define  $N_i = \{u \mid \text{dist}(u, N_0) = i\}$ . We call the sets  $N_i$  ( $i \geq 0$ ) the *layers* of  $G$ .

In the remainder, we consider  $N_0$  to be a fixed set of vertices. That is, we will update  $(G, L)$  by applying a number of propagation rules and doing some (polynomial) branching, but we will never delete the vertices of  $N_0$ . This will enable us to exploit the  $H$ -freeness of  $G$ .

We show the following two claims about layers (proofs omitted).

► **Claim 8.**  $V = N_0 \cup N_1 \cup N_2 \cup N_3$ .

► **Claim 9.**  $G[N_2 \cup N_3]$  is the disjoint union of complete graphs of size at most 3, each containing at least one vertex of  $N_2$  (and thus at most two vertices of  $N_3$ ).

We will now introduce a number of propagation rules, which run in polynomial time. We are going to apply these rules on  $G$  *exhaustively*, that is, until none of the rules can be applied anymore. Note that during this process some vertices of  $G$  may be deleted (due to Rules 2 and 2), but as mentioned we will ensure that we keep the vertices of  $N_0$ , while we may update the other sets  $N_i$  ( $i \geq 1$ ). We say that a propagation rule is *safe* if the new instance is a yes-instance of LIST 3-COLOURING if and only if the original instance is so.

**Rule 1. (no empty lists)** If  $L(u) = \emptyset$  for some  $u \in V$ , then return **no**.

**Rule 2. (not only lists of size 2)** If  $|L(u)| \leq 2$  for every  $u \in V$ , then apply Theorem 6.

**Rule 3. (connected graph)** If  $G$  is disconnected, then solve LIST 3-COLOURING on each instance  $(D, L_D)$ , where  $D$  is a connected component of  $G$  that does not contain  $N_0$  and  $L_D$  is the restriction of  $L$  to  $D$ . If  $D$  has no colouring respecting  $L_D$ , then return **no**; otherwise remove the vertices of  $D$  from  $G$ .

**Rule 4. (no coloured vertices)** If  $u \notin N_0$ ,  $|L(u)| = 1$  and  $L(u) \cap L(v) = \emptyset$  for all  $v \in N(u)$ , then remove  $u$  from  $G$ .

**Rule 5. (single colour propagation)** If  $u$  and  $v$  are adjacent,  $|L(u)| = 1$ , and  $L(u) \subseteq L(v)$ , then set  $L(v) := L(v) \setminus L(u)$ .

**Rule 6. (diamond colour propagation)** If  $u$  and  $v$  are adjacent and share two common neighbours  $x$  and  $y$  with  $L(x) \neq L(y)$ , then set  $L(x) := L(x) \cap L(y)$  and  $L(y) := L(x) \cap L(y)$ .

**Rule 7. (twin colour propagation)** If  $u$  and  $v$  are non-adjacent,  $N(u) \subseteq N(v)$ , and  $L(v) \subseteq L(u)$ , then set  $L(u) := L(v)$ .

**Rule 8. (triangle colour propagation)** If  $u, v, w$  form a triangle,  $|L(u) \cup L(v)| = 2$  and  $|L(w)| \geq 2$ , then set  $L(w) := L(w) \setminus (L(u) \cup L(v))$ , so  $|L(w)| \leq 1$ .

**Rule 9. (no free colours)** If  $|L(u) \setminus L(N(u))| \geq 1$  and  $|L(u)| \geq 2$  for some  $u \in V$ , then set  $L(u) := \{c\}$  for some  $c \in L(u) \setminus L(N(u))$ .

**Rule 10. (no small degrees)** If  $|L(u)| > |\deg(u)|$  for some  $u \in V \setminus N_0$ , then remove  $u$  from  $G$ .

As mentioned, our algorithm will branch at several stages to create a number of new but smaller instances, such that the original instance is a yes-instance if and only if at least one of the new instances is a yes-instance. Unless we explicitly state otherwise, we *implicitly* assume that Rules 2–2 are applied exhaustively immediately after we branch (see also Claim 10). If we apply Rule 2 or 2 on a new instance, then a no-answer means that we will discard the branch. So our algorithm will only return a no-answer for the original instance  $(G, L)$  if we discarded all branches. On the other hand, if we can apply Rule 2 on some new instance and obtain a yes-answer, then we can extend the obtained colouring to a colouring of  $G$  that respects  $L$ , simply by restoring all the already coloured vertices that were removed from the graph due to the rules. We will now state (without proof) Claim 10.

► **Claim 10.** *Rules 2–2 are safe and their exhaustive application takes polynomial time. Moreover, if we have not obtained a yes- or no-answer, then afterwards  $G$  is a connected  $(H, K_4)$ -free graph, such that  $V = N_0 \cup N_1 \cup N_2 \cup N_3$  and  $2 \leq |L(u)| \leq 3$  for every  $u \in V \setminus N_0$ .*

### Phase 1. Preprocessing $(G, L)$

In Phase 1 we will preprocess  $(G, L)$  using the above propagation rules. To start off the preprocessing we will branch via colouring the vertices of  $N_0$  in every possible way. By colouring a vertex  $u$ , we mean reducing the list of permissible colours to size exactly one. (When  $L(u) = \{c\}$ , we consider vertex coloured by colour  $c$ .) Thus, when we colour some vertex  $u$ , we always give  $u$  a colour from its list  $L(u)$ , moreover, when we colour more than one vertex we will always assign distinct colours to adjacent vertices.

#### Branching I. ( $O(1)$ branches)

We now consider all possible combinations of colours that can be assigned to the vertices in  $N_0$ . That is, we branch into at most  $3^7$  cases, in which  $v_1, \dots, v_7$  each received a colour from their list. We note that each branch leads to a smaller instance and that  $(G, L)$  is a yes-instance if and only if at least one of the new instances is a yes-instance. Hence, if we applied Rule 2 in some branch, then we discard the branch. If we applied Rule 2 and obtained a no-answer, then we discard the branch as well. If we obtained a yes-answer, then we are done. Otherwise we continue by considering each remaining branch separately. For each remaining branch, we denote the resulting smaller instance by  $(G, L)$  again.

We will now introduce a new rule, namely Rule 2. We apply Rule 2 together with the other rules. That is, we now apply Rules 2–2 exhaustively. However, each time we apply Rule 2 we first ensure that Rules 2–2 have been applied exhaustively.

**Rule 11. ( $N_3$ -reduction)** If  $u$  and  $v$  are in  $N_3$  and are adjacent, then remove  $u$  and  $v$  from  $G$ . We state (without proofs) the following claims.

► **Claim 11.** *Rule 2, applied after exhaustive application of Rules 2–2, is safe and takes polynomial time. Moreover, afterwards  $G$  is a connected  $(H, K_4)$ -free graph, such that  $V = N_0 \cup N_1 \cup N_2 \cup N_3$  and  $2 \leq |L(u)| \leq 3$  for every  $u \in V \setminus N_0$ .*



► **Claim 12.** *The set  $N_3$  is independent, and moreover, each vertex  $u \in N_3$  has  $|L(u)| = 2$  and exactly two neighbours in  $N_2$  which are adjacent.*

The following claim follows immediately from Claims 9 and 12.

► **Claim 13.** *Every connected component  $D$  of  $G[N_2 \cup N_3]$  is a complete graph with either  $|D| \leq 2$  and  $D \subseteq N_2$ , or  $|D| = 3$  and  $|D \cap N_3| \leq 1$ .*

The following claim (proof omitted) describes the location of the vertices with a list of size 3.

► **Claim 14.** *For every  $u \in V$ , if  $|L(u)| = 3$ , then  $u \in N_2$ .*

We will now show how to branch in order to reduce the lists of the vertices  $u \in N_2$  with  $|L(u)| = 3$  by at least one colour. We formalize this approach in the following definition.

► **Definition 15 (Active vertices).** A vertex  $u \in N_2$  and its neighbours in  $N_1$  are called *active* if  $|L(u)| = 3$ . Let  $A$  be the set of all active vertices. Let  $A_1 = A \cap N_1$  and  $A_2 = A \cap N_2$ . We *deactivate* a vertex  $u \in A_2$  if we reduce the list  $L(u)$  by at least one colour. We *deactivate* a vertex  $w \in A_1$  by deactivating all its neighbours in  $A_2$ .

Note that every vertex  $w \in A_1$  has  $|L(w)| = 2$  by Rule 2 applied on the vertices of  $N_0$ . Hence, if we reduce  $L(w)$  by one colour, all neighbours of  $w$  in  $A_2$  become deactivated by Rule 2, and  $w$  is removed by Rule 2. For  $1 \leq i \leq j \leq 7$ , we let  $A(i, j) \subseteq A_1$  be the set of active neighbours of  $v_i$  that are not adjacent to  $v_j$  and similarly, we let  $A(j, i) \subseteq A_1$  be the set of active neighbours of  $v_j$  that are not adjacent to  $v_i$ .

## Phase 2. Reduce the number of distinct sets $A(i, j)$

We will now branch into  $O(n^{45})$  smaller instances such that  $(G, L)$  is a yes-instance of LIST 3-COLOURING if and only if at least one of these new instances is a yes-instance. Each new instance will have the following property:

(P) for  $1 \leq i \leq j \leq 7$  with  $j - i \geq 2$ , either  $A(i, j) = \emptyset$  or  $A(j, i) = \emptyset$ .

**Branching II.**  $(O(n^{3 \cdot (\binom{7}{2} - 6)})) = O(n^{45})$  branches

Consider two vertices  $v_i$  and  $v_j$  with  $1 \leq i \leq j \leq 7$  and  $j - i \geq 2$ . Assume without loss of generality that  $v_i$  is coloured 3 and that  $v_j$  is coloured either 1 or 3. Hence, every  $w \in A(i, j)$  has  $L(w) = \{1, 2\}$ , whereas every  $w \in A(j, i)$  has  $L(w) = \{2, q\}$  for  $q \in \{1, 3\}$ . We branch as follows. We consider all possibilities where at most one vertex of  $A(i, j)$  receives colour 2 (and all other vertices of  $A(i, j)$  receive colour 1) and all possibilities where we choose two vertices from  $A(i, j)$  to receive colour 2. This leads to  $O(n) + O(n^2) = O(n^2)$  branches. In the branches where at most one vertex of  $A(i, j)$  receives colour 2, every vertex of  $A(i, j)$  will be deactivated. So Property (P) is satisfied for  $i$  and  $j$ .

Now consider the branches where two vertices  $x_1, x_2$  of  $A(i, j)$  both received colour 2. We update  $A(j, i)$  accordingly. In particular, afterwards no vertex in  $A(j, i)$  is adjacent to  $x_1$  or  $x_2$ , as 2 is a colour in the list of each vertex of  $A(j, i)$ . We now do some further branching for those branches where  $A(j, i) \neq \emptyset$ . We consider the possibility where each vertex of  $N(A(j, i)) \cap A_2$  is given the colour of  $v_j$  and all possibilities where we choose one vertex in  $N(A(j, i)) \cap A_2$  to receive a colour different from the colour of  $v_j$  (we consider both options to colour such a vertex). This leads to  $O(n)$  branches. In the first branch, every vertex of  $A(j, i)$  will be deactivated. So Property (P) is satisfied for  $i$  and  $j$ .

Now consider a branch where a vertex  $u \in N(A(j, i)) \cap A_2$  receives a colour different from the colour of  $v_j$ . We will show that also in this case every vertex of  $A(j, i)$  will be deactivated.



For contradiction, assume that  $A(j, i)$  contains a vertex  $w$  that is not deactivated after colouring  $u$ . As  $u$  was in  $N(A(j, i)) \cap A_2$ , we find that  $u$  had a neighbour  $w' \in A(j, i)$ . As  $u$  is coloured with a colour different from the colour of  $v_j$ , the size of  $L(w')$  is reduced by one (due to Rule 2). Hence  $w'$  got deactivated after colouring  $u$ , and thus  $w' \neq w$ . As  $w$  is still active,  $w$  has a neighbour  $u' \in A_2$ . As  $u'$  and  $w$  are still active,  $u'$  and  $w$  are not adjacent to  $w'$  or  $u$ . Hence,  $u, w', v_j, w, u'$  induce a  $P_5$  in  $G$ . As  $x_1$  and  $x_2$  both received colour 2, we find that  $x_1$  and  $x_2$  are not adjacent to each other. Hence,  $x_1, v_i, x_2$  induce a  $P_3$  in  $G$ . Recall that all vertices of  $A(j, i)$ , so also  $w$  and  $w'$ , are not adjacent to  $x_1$  or  $x_2$ . As  $u$  and  $u'$  were still active after colouring  $x_1$  and  $x_2$ , we find that  $u$  and  $u'$  are not adjacent to  $x_1$  or  $x_2$  either. By definition of  $A(j, i)$ ,  $w$  and  $w'$  are not adjacent to  $v_i$ . By definition of  $A(i, j)$ ,  $x_1$  and  $x_2$  are not adjacent to  $v_j$ . Moreover,  $v_i$  and  $v_j$  are non-adjacent, as  $j - i \geq 2$ . We conclude that  $G$  contains an induced  $P_3 + P_5$ , namely with vertex set  $\{x_1, v_i, x_2\} \cup \{u, w', v_j, w, u'\}$ , a contradiction. Hence, every vertex of  $A(j, i)$  is deactivated. So Property **(P)** is satisfied for  $i$  and  $j$  also for these branches.

Finally by recursive application of the above procedure for all pairs  $v_i, v_j$  such that  $1 \leq i \leq j \leq 7$  and  $j - i \geq 2$  we get a graph satisfying Property **(P)**.

We now consider each resulting instance from Branching II. We denote such an instance by  $(G, L)$  again. Note that vertices from  $N_2$  may now belong to  $N_3$ , as their neighbours in  $N_1$  may have been removed due to the branching. The exhaustive application of Rules 2–2 preserves **(P)** (where we apply Rule 2 only after applying Rules 2–2 exhaustively). Hence  $(G, L)$  satisfies **(P)**.

We observe that if two vertices in  $A_1$  have a different list, then they must be adjacent to different vertices of  $N_0$ . Hence, by Property **(P)**, at most two lists of  $\{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$  can occur as lists of vertices of  $A_1$ . Without loss of generality this leads to two cases: either every vertex of  $A_1$  has list  $\{1, 2\}$  or  $\{1, 3\}$  and both lists occur on  $A_1$ ; or every vertex of  $A_1$  has list  $\{1, 2\}$  only. In the next phase of our algorithm we reduce, via some further branching, every instance of the first case to a polynomial number of smaller instances of the second case.

### Phase 3. Reduce to the case where vertices of $A_1$ have the same list

Recall that we assume that every vertex of  $A_1$  has list  $\{1, 2\}$  or  $\{1, 3\}$ . In this phase we deal with the case when both types of lists occur in  $A_1$ . We first show, without proof, the following two claims.

► **Claim 16.** *Let  $i \in \{1, 3, 5, 7\}$ . Then every vertex from  $A_1 \cap N(v_i)$  is adjacent to some vertex  $v_j$  with  $j \notin \{i - 1, i, i + 1\}$ .*

► **Claim 17.** *It holds that  $N(A_1) \cap N_0 = \{v_{i-1}, v_i, v_{i+1}\}$  for some  $2 \leq i \leq 6$ . Moreover, we may assume without loss of generality that  $v_{i-1}$  and  $v_{i+1}$  have colour 3 and both are adjacent to all vertices of  $A_1$  with list  $\{1, 2\}$ , whereas  $v_i$  has colour 2 and is adjacent to all vertices of  $A_1$  with list  $\{1, 3\}$ .*

By Claim 17, we can partition the set  $A_1$  into two (non-empty) sets  $X_{1,2}$  and  $X_{1,3}$ , where  $X_{1,2}$  is the set of vertices in  $A_1$  with list  $\{1, 2\}$  whose only neighbours in  $N_0$  are  $v_{i-1}$  and  $v_{i+1}$  (which both have colour 3) and  $X_{1,3}$  is the set of vertices in  $A_1$  with list  $\{1, 3\}$  whose only neighbour in  $N_0$  is  $v_i$  (which has colour 2).

Our goal is to show that we can branch into at most  $O(n^2)$  smaller instances, in which either  $X_{1,2} = \emptyset$  or  $X_{1,3} = \emptyset$ , such that  $(G, L)$  is a yes-instance of LIST 3-COLOURING if and only if at least one of these smaller instances is a yes-instance. Then afterwards it suffices to

show how to deal with the case where all vertices in  $A_1$  have the same list in polynomial time; this will be done in Phase 4 of the algorithm. We start with the following  $O(n)$  branching procedure (in each of the branches we may do some further  $O(n)$  branching later on).

**Branching III.** ( $O(n)$  branches)

We branch by considering the possibility of giving each vertex in  $X_{1,2}$  colour 2 and all possibilities of choosing a vertex in  $X_{1,2}$  and giving it colour 1. This leads to  $O(n)$  branches. In the first branch we obtain  $X_{1,2} = \emptyset$ . Hence we can start Phase 4 for this branch. We now consider every branch in which  $X_{1,2}$  and  $X_{1,3}$  are both nonempty. For each such branch we will create  $O(n)$  smaller instances of LIST 3-COLOURING, where  $X_{1,3} = \emptyset$ , such that  $(G, L)$  is a yes-instance of LIST 3-COLOURING if and only if at least one of the new instances is a yes-instance.

Let  $w \in X_{1,2}$  be the vertex that was given colour 1 in such a branch. Although by Rule 2 vertex  $w$  will need to be removed from  $G$ , we make an exception by temporarily keeping  $w$  after we coloured it. The reason is that the presence of  $w$  will be helpful for analysing the structure of  $(G, L)$  after Rules 2–2 have been applied exhaustively (where we apply Rule 2 only after applying Rules 2–2 exhaustively). In order to do this, we first show the following three claims (proofs omitted).

► **Claim 18.** *Vertex  $w$  is not adjacent to any vertex in  $A_2 \cup X_{1,2} \cup X_{1,3}$ .*

► **Claim 19.** *The graph  $G[X_{1,3} \cup (N(X_{1,3}) \cap A_2) \cup N_3]$  is the disjoint union of one or more complete graphs, each of which consists of either one vertex of  $X_{1,3}$  and at most two vertices of  $A_2$ , or one vertex of  $N_3$ .*

► **Claim 20.** *For every pair of adjacent vertices  $s, t$  with  $s \in A_2$  and  $t \in N_2$ , either  $t$  is adjacent to  $w$ , or  $N(s) \cap X_{1,3} \subseteq N(t)$ .*

We now continue as follows. Recall that  $X_{1,3} \neq \emptyset$ . Hence there exists a vertex  $s \in A_2$  that has a neighbour  $r \in X_{1,3}$ . As  $s \in A_2$ , we have that  $|L(s)| = 3$ . Then, by Rule 2, we find that  $s$  has at least two neighbours  $t$  and  $t'$  not equal to  $r$ . By Claim 19, we find that neither  $t$  nor  $t'$  belongs to  $X_{1,3} \cup N_3$ . We are going to fix an induced 3-vertex path  $P^s$  of  $G$ , over which we will branch, in the following way.

If  $t$  and  $t'$  are not adjacent, then we let  $P^s$  be the induced path in  $G$  with vertices  $t, s, t'$  in that order. Suppose that  $t$  and  $t'$  are adjacent. As  $G$  is  $K_4$ -free and  $s$  is adjacent to  $r, t, t'$ , at least one of  $t, t'$  is not adjacent to  $r$ . We may assume without loss of generality that  $t$  is not adjacent to  $r$ .

First assume that  $t \in N_2$ . Recall that  $s$  has a neighbour in  $X_{1,3}$ , namely  $r$ , and that  $r$  is not adjacent to  $t$ . We then find that  $t$  must be adjacent to  $w$  by Claim 20. As  $s \in A_2$ , we find that  $s$  is not adjacent to  $w$  by Claim 18. In this case we let  $P^s$  be the induced path in  $G$  with vertices  $s, t, w$  in that order.

Now assume that  $t \notin N_2$ . Recall that  $t \notin N_3$ . Hence,  $t$  must be in  $N_1$ . Then, as  $t \notin X_{1,3}$  but  $t$  is adjacent to a vertex in  $A_2$ , namely  $s$ , we find that  $t \in X_{1,2}$ . Recall that  $t' \notin X_{1,3}$ . If  $t' \in N_1$  then the fact that  $t' \notin X_{1,3}$ , combined with the fact that  $t'$  is adjacent to  $s \in A_2$ , implies that  $t' \in X_{1,2}$ . However, by Rule 2 applied on  $s, t, t'$ , vertex  $s$  would have a list of size 1 instead of size 3, a contradiction. Hence,  $t' \notin N_1$ . As  $t' \notin N_3$ , this means that  $t' \in N_2$ . If  $t'$  is adjacent to  $r$ , then  $t \in X_{1,2}$  with  $L(t) = \{1, 2\}$  and  $r \in X_{1,3}$  with  $L(r) = \{1, 3\}$  would have the same lists by Rule 2 applied on  $r, s, t, t'$ , a contradiction. Hence  $t'$  is not adjacent to  $r$ . Then, by Claim 20, we find that  $t'$  must be adjacent to  $w$ . Note that  $s$  is not adjacent to  $w$  due to Claim 18. In this case we let  $P^s$  be the induced path in  $G$  with vertices  $s, t', w$

in that order. We conclude that either  $P^s = tst'$  or  $P^s = stw$  or  $P^s = st'w$ . We are now ready to apply two more rounds of branching.

**Branching IV.** ( $O(n)$  branches)

We branch by considering the possibility of removing colour 2 from the list of each vertex in  $N(X_{1,3}) \cap A_2$  and all possibilities of choosing a vertex in  $N(X_{1,3}) \cap A_2$  and giving it colour 2. In the branch where we removed colour 2 from the list of every vertex in  $N(X_{1,3}) \cap A_2$ , we obtain that  $X_{1,3} = \emptyset$ . Hence for that branch we can enter Phase 4. Now consider a branch where we gave some vertex  $s \in N(X_{1,3}) \cap A_2$  colour 2. Let  $P^s = tst'$  or  $P^s = stw$  or  $P^s = st'w$ . We do some further branching by considering all possibilities of colouring the vertices of  $P^s$  that are not equal to the already coloured vertices  $s$  and  $w$  (should  $w$  be a vertex of  $P^s$ ) and all possibilities of giving a colour to the vertex from  $N(s) \cap X_{1,3}$  (recall that by Claim 19,  $|N(s) \cap X_{1,3}| = 1$ ). This leads to a total of  $O(n)$  branches. We claim that in both branches,  $|X_{1,3}|$  has reduced to at most 1 (proof omitted).

**Branching V.** ( $O(1)$  branches)

We branch by considering both possibilities of colouring the unique vertex of  $X_{1,3}$ . This leads to two new but smaller instances of LIST 3-COLOURING, in each of which the set  $X_{1,3} = \emptyset$ . Hence, our algorithm can enter Phase 4.

**Phase 4. Reduce to a set of instances of 2-List Colouring**

Recall that in this stage of our algorithm we have an instance  $(G, L)$  in which every vertex of  $A_1$  has the same list, say  $\{1, 2\}$ . As  $G$  is  $(P_2 + P_5)$ -free,  $G[N_2 \cup N_3]$  is an independent set; otherwise two adjacent vertices of  $N_2 \cup N_3$  form, together with  $v_1, \dots, v_5$ , an induced  $P_2 + P_5$ . Hence, we can safely colour each vertex in  $A_2$  with colour 3, and afterwards we may apply Theorem 6.

The correctness of our algorithm follows from the description. The branching in the five stages (Branching I-V), yields a total number of  $O(n^{47})$  branches and each branch we created takes polynomial time to process. Hence, the running time of our algorithm is polynomial.  $\blacktriangleleft$

► **Remark.** Except for Phase 4 of our algorithm, all arguments in our proof hold for  $(P_3 + P_5)$ -free graphs. The difficulty in Phase 4 is that in contrary to the previous phases we cannot use the vertices from  $N_0$  to find an induced  $P_3 + P_5$  and therefore obtain the contradiction.

### 3 The Hardness Result

We show that 5-COLOURING is NP-complete for  $(P_3 + P_5)$ -free graphs by reducing from the NP-complete problem [32] NOT-ALL-EQUAL 3-SATISFIABILITY with positive literals only, defined as follows: given a set  $X = \{x_1, x_2, \dots, x_n\}$  of logical variables and a set  $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$  of 3-literal clauses over  $X$  in which all literals are positive, is there a truth assignment for  $X$  such that each clause contains at least one true literal and at least one false literal? We call such a truth assignment *satisfying*.

► **Theorem 4 (restated).** 5-COLOURING is NP-complete for  $(P_3 + P_5)$ -free graphs.

**Proof.** Proof Sketch. From a given instance  $(\mathcal{C}, X)$  of NOT-ALL-EQUAL 3-SATISFIABILITY with positive literals only, we first construct a graph  $G$  with a list assignment  $L$ . For each  $x_i \in X$  we introduce two vertices  $x_i$  and  $\bar{x}_i$ , which we make adjacent to each other. We say that  $x_i$  and  $\bar{x}_i$  are of  $x$ -type. We set  $L(x_i) = L(\bar{x}_i) = \{4, 5\}$ . For each  $C_j \in \mathcal{C}$  we introduce

a vertex  $C_j$  and a vertex  $C'_j$  called the *copy* of  $C_j$ . We say that  $C_j$  and  $C'_j$  are of  $C$ -type. We set  $L(C_j) = L(C'_j) = \{1, 2, 3\}$ . We add an edge between each  $x$ -type vertex and each  $C$ -type vertex. For each  $C_j \in \mathcal{C}$  we do as follows. We fix an arbitrary order of the literals in  $C_j$ . Say  $C_j = \{x_g, x_h, x_i\}$  in that order. Then we add six vertices  $a_{g,j}, a_{h,j}, a_{i,j}, a'_{g,j}, a'_{h,j}, a'_{i,j}$  and edges  $x_g a_{g,j}, a_{g,j} C_j, x_h a_{h,j}, a_{h,j} C_j, x_i a_{i,j}, a_{i,j} C_j$  and also edges  $\bar{x}_g a'_{g,j}, a'_{g,j} C'_j, \bar{x}_h a'_{h,j}, a'_{h,j} C'_j, \bar{x}_i a'_{i,j}, a'_{i,j} C'_j$ . We say that  $a_{g,j}, a_{h,j}, a_{i,j}, a'_{g,j}, a'_{h,j}, a'_{i,j}$  are of  $a$ -type. We set  $L(a_{g,j}) = L(a'_{g,j}) = \{1, 4\}$ ,  $L(a_{h,j}) = L(a'_{h,j}) = \{2, 4\}$  and  $L(a_{i,j}) = L(a'_{i,j}) = \{3, 4\}$ .

We now extend  $G$  into a graph  $G'$  by adding a clique consisting of five new vertices  $k_1, \dots, k_5$ , which we say are of  $k$ -type, and by adding an edge between a vertex  $k_\ell$  and a vertex  $u \in V(G)$  if and only if  $\ell \notin L(u)$ . We can show that  $(\mathcal{C}, X)$  has a satisfying truth assignment if and only if  $G'$  has a 5-colouring, and moreover that  $G'$  is  $(P_3 + P_5)$ -free (proof omitted). As 5-COLOURING belongs to NP, this proves the theorem. ◀

## 4 Conclusions

By solving two new cases we completed the complexity classifications of 3-COLOURING and LIST 3-COLOURING on  $H$ -free graphs for graphs  $H$  up to seven vertices. We showed that both problems become polynomial-time solvable if  $H$  is a linear forest, while they stay NP-complete in all other cases. Recall that  $k$ -COLOURING ( $k \geq 3$ ) is NP-complete on  $H$ -free graphs whenever  $H$  is not a linear forest. For the case where  $H$  is a linear forest, our new NP-hardness result for 5-COLOURING for  $(P_3 + P_5)$ -free graphs bounds, together with the known NP-hardness results of [20] for 4-COLOURING for  $P_7$ -free graphs and 5-COLOURING for  $P_6$ -free graphs, the number of open cases of  $k$ -COLOURING from above.

For future research we aim to extend our results. In fact we still do not know if there exists a linear forest  $H$  such that 3-COLOURING is NP-complete for  $H$ -free graphs. This is, however, a notorious open problem studied in many papers; for a recent discussion see [16]. It is also open for LIST 3-COLOURING, where an affirmative answer to one of the two problems yields an affirmative answer to the other one [15]. For  $k \geq 4$ , we emphasize that all open cases involve linear forests  $H$  whose connected components are small. For instance, if  $H$  has at most six vertices, then the polynomial-time algorithm for 4-PRECOLOURING EXTENSION on  $P_6$ -free graphs [7, 8] implies that there are only three graphs  $H$  with  $|V(H)| \leq 6$  for which we do not know the complexity of 4 COLOURING on  $H$ -free graphs, namely  $H \in \{P_1 + P_2 + P_3, P_2 + P_4, 2P_3\}$  (see [14]).

The main difficulty to extend the known complexity results is that hereditary graph classes characterized by a forbidden induced linear forest are still not sufficiently well understood due to their rich structure. We need a better understanding of these graph classes to make further progress on a wide range of problems. For example, INDEPENDENT SET is polynomial-time solvable for  $P_6$ -free graphs [17], but it is not known if there exists a linear forest  $H$  such that it is NP-complete for  $H$ -free graphs. A similar situation holds for ODD CYCLE TRANSVERSAL and FEEDBACK VERTEX SET and many other problems; see [1] for a survey.

---

## References

- 1 Marthe Bonamy, Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, and Daniël Paulusma. Independent feedback vertex set for  $P_5$ -free graphs. *Algorithmica*, to appear.
- 2 Flavia Bonomo, Maria Chudnovsky, Peter Maceli, Oliver Schaudt, Maya Stein, and Mingxian Zhong. Three-coloring and list three-coloring of graphs without induced paths on seven vertices. *Combinatorica*, (in press).

- 3 Hajo Broersma, Fedor V. Fomin, Petr A. Golovach, and Daniël Paulusma. Three complexity results on coloring  $P_k$ -free graphs. *European Journal of Combinatorics*, 34(3):609–619, 2013.
- 4 Hajo Broersma, Petr A. Golovach, Daniël Paulusma, and Jian Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *TCS*, 414(1):9–19, 2012.
- 5 Maria Chudnovsky. Coloring graphs with forbidden induced subgraphs. *Proc. ICM 2014*, IV:291–302, 2014.
- 6 Maria Chudnovsky, Peter Maceli, Juraj Stacho, and Mingxian Zhong. 4-Coloring  $P_6$ -free graphs with no induced 5-cycles. *Journal of Graph Theory*, 84(3):262–285, 2017.
- 7 Maria Chudnovsky, Sophie Spirkl, and Mingxian Zhong. Four-coloring  $P_6$ -free graphs. I. Extending an excellent precoloring. *CoRR*, 1802.02282, 2018.
- 8 Maria Chudnovsky, Sophie Spirkl, and Mingxian Zhong. Four-coloring  $P_6$ -free graphs. II. Finding an excellent precoloring. *CoRR*, 1802.02283, 2018.
- 9 Maria Chudnovsky and Juraj Stacho. 3-colorable subclasses of  $P_8$ -free graphs. *Man.*, 2017.
- 10 Jean-François Couturier, Petr A. Golovach, Dieter Kratsch, and Daniël Paulusma. List Coloring in the Absence of a Linear Forest. *Algorithmica*, 71(1):21–35, 2015.
- 11 Konrad K. Dabrowski and Daniël Paulusma. On colouring  $(2P_2, H)$ -free and  $(P_5, H)$ -free graphs. *Information Processing Letters*, 131:26–32, 2018.
- 12 Keith Edwards. The complexity of colouring problems on dense graphs. *TCS*, 43:337–343, 1986.
- 13 Thomas Emden-Weinert, Stefan Hougardy, and Bernd Kreuter. Uniquely Colourable Graphs and the Hardness of Colouring Graphs of Large Girth. *Combinatorics, Probability and Computing*, 7(04):375–386, 1998.
- 14 Petr A. Golovach, Matthew Johnson, Daniël Paulusma, and Jian Song. A Survey on the Computational Complexity of Colouring Graphs with Forbidden Subgraphs. *Journal of Graph Theory*, 84(4):331–363, 2017.
- 15 Petr A. Golovach, Daniël Paulusma, and Jian Song. Closing complexity gaps for coloring problems on  $H$ -free graphs. *Information and Computation*, 237:204–214, 2014.
- 16 Carla Groenland, Karolina Okrasa, Paweł Rzażewski, Alex Scott, Paul Seymour, and Sophie Spirkl.  $H$ -colouring  $P_t$ -free graphs in subexponential time. *CoRR*, 1803.05396, 2018.
- 17 Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. Polynomial-time algorithm for Maximum Weight Independent Set on  $P_6$ -free graphs. *CoRR*, 1707.05491, 2017.
- 18 Chinh T. Hoàng, Marcin Kamiński, Vadim V. Lozin, Joe Sawada, and Xiao Shu. Deciding  $k$ -Colorability of  $P_5$ -Free Graphs in Polynomial Time. *Algorithmica*, 57(1):74–81, 2010.
- 19 Ian Holyer. The NP-Completeness of Edge-Coloring. *SIAM J Comput.*, 10(4):718–720, 1981.
- 20 Shenwei Huang. Improved complexity results on  $k$ -coloring  $P_t$ -free graphs. *European Journal of Combinatorics*, 51:336–346, 2016.
- 21 Tommy R. Jensen and Bjarne Toft. *Graph coloring problems*. John Wiley & Sons, 1995.
- 22 Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring  $(P_r + P_s)$ -Free Graphs. *CoRR*, 2018. [arXiv:1804.11091v2](https://arxiv.org/abs/1804.11091v2).
- 23 Daniel Král', Jan Kratochvíl, Zolt Tuza, and Gerhard J. Woeginger. Complexity of Coloring Graphs without Forbidden Induced Subgraphs. *Proc. WG 2001, LNCS*, 2204:254–262, 2001.
- 24 Jan Kratochvíl, Zolt Tuza, and Margit Voigt. New trends in the theory of graph colorings: choosability and list coloring. *Proc. DIMATIA-DIMACS Conference*, 49:183–197, 1999.
- 25 Van Bang Le, Bert Randerath, and Ingo Schiermeyer. On the complexity of 4-coloring graphs without long induced paths. *TCS*, 389(1-2):330–335, 2007.

- 26 Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44, 1983.
- 27 László Lovász. Coverings and coloring of hypergraphs. *Congr. Numer.*, VIII:3–12, 1973.
- 28 Daniël Paulusma. Open problems on graph coloring for special graph classes. *Proc. WG 2015, LNCS*, 9224:16–30, 2015.
- 29 Bert Randerath and Ingo Schiermeyer. 3-Colorability in P for  $P_6$ -free graphs. *Discrete Applied Mathematics*, 136(2-3):299–313, 2004.
- 30 Bert Randerath and Ingo Schiermeyer. Vertex Colouring and Forbidden Subgraphs – A Survey. *Graphs and Combinatorics*, 20(1):1–40, 2004.
- 31 Bert Randerath, Ingo Schiermeyer, and Meike Tewes. Three-colourability and forbidden subgraphs. II: polynomial algorithms. *Discrete Mathematics*, 251(1–3):137–153, 2002.
- 32 Thomas J. Schaefer. The Complexity of Satisfiability Problems. *STOC*, pages 216–226, 1978.
- 33 Zsolt Tuza. Graph colorings with local constraints - a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997.
- 34 Gerhard J. Woeginger and Jiří Sgall. The complexity of coloring graphs without long induced paths. *Acta Cybernetica*, 15(1):107–117, 2001.